Real-time Simulation of Distributed Control Systems: The example of Functional Electrical Stimulation

Daniel Simon, David Andreu

Real-time Simulation of Distributed Control Systems : the example of Functional Electrical Stimulation

Daniel Simon[1], David Andreu[2]
[1] *CAMIN team, INRIA, LIRMM Bat 5 CC05 017, 860 Rue Saint Priest, Montpellier Cedex 5, France*
[2] *CAMIN team, Univ. Montpellier, LIRMM Bat 5 CC05 017, 860 Rue Saint Priest, Montpellier Cedex 5, France*
daniel.simon@inria.fr, andreu@lirmm.fr

Keywords: Hybrid simulation, Real-time control, Control architecture validation

Abstract: The paper presents a real-time open software simulation framework, dedicated to the analysis of control systems deployed over distributed execution resources and wireless links. It is able to consistently simulate in parallel the numerical devices (real-time tasks and communication links) and the evolution of the controlled continuous time plant. It is applied to foresee future enhancements of a Functional Electrical Stimulation (FES) system used in therapy for rehabilitation or substitution for disabled people. It is a distributed control system using electrodes to interface a digital control system with livings. Hence the whole system gathers continuous-time (muscles and nerves) and discrete-time (controllers and wireless links) components. During the design process, realistic simulation remains a precious tool ahead of real experiments to check without danger that the implementation matches the functional and safety requirements. The simulation tool is especially devoted to the joint design and analysis of control loops and real-time features.

1 FROM CONTROL DESIGN TO REAL-TIME EXPERIMENTS

The design and validation of complex systems, like cyber-physical systems which include both physical and computational devices, is costly, time consuming and needs knowledge and cooperation of different disciplines (Ben Khaled-El Feki, 2014). For example, medical robotics belong to such category and require the coordinated design of both bio-mechanical, electrical, and chemical models (from a physical point of view) and many-sided controllers (from a computational point of view).

Simulations, handling both the components and their interaction, are needed from the early stage to speed-up the design, development and validation phases, especially when real experiments are costly or dangerous for human beings.

1.1 Simulation needs

Understanding and modeling the influence of an implementation (support system) on the QoC (Quality of control) is a challenging objective in control/computing co-design process. Real-time properties (task response times) and the network Quality of Service (QoS) influence the controlled system properties and QoC. However real-life size problems, involving non-linear systems and uncertain components, still escape from a purely theoretic framework. In this design process, simulation is an indisputable step between concept design and prototype validation.

Models are needed to describe the mechatronic continuous system, to compute the discrete time control laws and diagnosers, and to handle the network behavior. As usual, choosing the right model results of a trade-off between complexity and fidelity. Indeed several models with different granularity are used during the design and validation process, from fast prototyping until real-time simulation and implementation.

1.2 Numerical simulation of control process

The main purpose of the numerical simulation is to approximate as faithfully as possible the behavior of the complex dynamic system. In other words, bounding the simulation errors is an important goal of numerical simulations so that the designers can be confident with the prediction.

A distinctive feature of real-time control systems is that they are *hybrid*. Indeed, they gather components with different time nature. Digital components,

such as CPUs and networks, run over discrete time scale while handling real-time tasks and communication packets. Digital components can be easily executed on the CPUs, using the multitasking capabilities of the simulation cores to simulate the concurrent nature of the distributed control system.

On the other hand, the physical components of the control system – such as actuators, sensors and mechatronic parts – are continuous time components which can be described by some kind of non-linear differential equations. As real system are too complex to lead to explicit solutions of their dynamic behavior, they must be integrated on computers using a numerical solver. Numerical integration is still an active field in numerical analysis, and provides many effective numerical solvers and associated software suites, able to solve the continuous models either using time discretization or state quantization (Ben Khaled-El Feki, 2014).

Finally, all the components of the simulated system must be carefully interconnected, and their time scales must be meshed to provide consistent simulation results over the different time representations and scales. Both modeling and numerical integration deal with approximations, hence it is first needed to find a satisfactory trade-off between the simulation speed and precision.

Real-time simulation needs to consider two time scales :

- Real-time : it is the time reference of the real physical system that is modeled and simulated, e.g. as measured by an absolute wall clock;
- Simulated time : it is the time elapsed during the execution of the simulation that can be measured by the numerical integrator clock (i.e., by the accumulation of integration steps).

To be consistent, the simulated time and real-time must be carefully meshed to meet at some precise points (Ben Khaled-El Feki, 2014).

Several simulation steps of growing accuracy and fidelity to the real implementation can be processed, from basic functional investigation in continuous time until hardware-in-the-loop setups including parts of the final components, just prior to real tests.

Model-in-the-loop A first step is the choice of a control structure, based on a model of the plant and on the control objective. The control toolbox now contains many tools to state and solve a large variety of control problems applied to various plants. Besides control theory, preliminary design and tests often rely on simulation tools such as Scilab/Scicos or Matlab/Simulink.

Software-in-the-loop As for MIL, the SIL phase considers only simulated (i.e. software) elements, but it takes into account the actual production code, including implementation related constraints such as fixed-point computations and memory size limits.

Hardware-in-the-loop After validating the software in a real-time SIL simulation, the production hardware can be checked in a real-time Hardware-In-the-Loop (HIL) simulation. A HIL simulation is made of mixed simulated and real components, which means that some real components –usually the physical control target– are replaced by their software avatar.

The next sections are devoted to the development of a simulation framework dedicated to the case of a distributed electrical functional electrical system.

2 THE FES FRAMEWORK

Functional electrical stimulation (FES) is one of existing rehabilitation techniques to restore lost motor functions for motor-impaired people. For example, in case of spinal cord injuries, the natural pathways between the central nervous system (CNS) and peripheral nerves are open under the lesion level. Therefore such lesions forbid both the activation of movements through motor nerves and the collection of sensory information by the CNS.

In FES systems, electrodes are connected to nerves or muscles and generate electrical pulses to induce contractions of muscles. Similar electrodes can be used to collect activity signals from muscles (ElectroMyoGrams EMG) or from nerves (ElectroNeuroGrams ENG). FES is primarily used for rehabilitation of functions for people with disabilities. Until now FES is mainly used in open-loop mode, where the therapist selects predefined currents patterns (e.g., frequency, intensity) according to the desired effect on the patient. It is argued, e.g. (Zhang et al., 2013), that closed-loop FES is necessary to safely control and coordinate the numerous electrodes and muscles needed to generate complex movements such as walking.

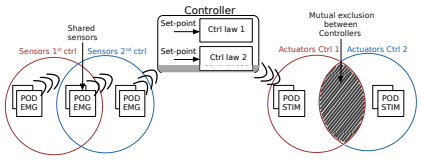

Figure 1: Feedback FES

In that case (Figure 1), the stimulation controller is fed back by various sensors, such as limbs joints

angles, IMUs providing accelerations, EMG signals... These signals are used by feedback controllers to accurately control the artificially actuated limbs.

Indeed it is well known that feedback control provides adaptability w.r.t. varying operation conditions and robustness w.r.t. the uncertainties spoiling the control loops components.

Compared with mechanical devices, the large uncertainties and variability of livings make them especially difficult to be accurately modeled and safely controlled. Conversely, considering humans in the loop, the safety of the technology must be validated, e.g. formally assessed, to allow for its certification and usage in everyday life on a large scale.

2.1 The Phenix Liberty FES system

The Phenix Liberty FES system has been developed along a partnership between the Demar research team and the Vivaltis company [1]. It is devoted to external stimulation, i.e. the electrodes are glued on the skin, facing the target muscle or nerve. Compared with its competitors, a distinctive feature of the system is its wireless implementation. Considering the multiple sensors and actuators needed to control complex motion, distributed systems over wireless links comply with mobility constraints, leading to acceptance from the human users. The marketed system executes a preset stimulation sequence in open-loop. Although it is fluently used to correct, e.g., foot drop, finite-state controllers typically are not adequate to adapt stimulation patterns for frequent changes of walking speed. FES also fastly involves muscular fatique, thus needed an on-line detection and adaptation of the stimulation profiles. Current research aims to design and implement closed-loop FES using this system.

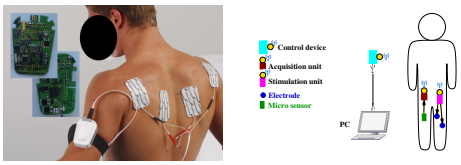

Figure 2: The Phenix FES system

The main components are :

- A main PC is used as a end-users (i.e. a therapist) interface, to select the stimulation pattern out of a library of pre-defined profiles, and to provide the gateway with the controller using an USB link;
- The *control device* manages the communications with the distributed stimulation units through the wireless network running the protocol stack. For future feedback controlled stimulation it will also run the real-time control loops. The controller board is equipped with a Freescale MC1322X micro-controller and an integrated IEEE802.15.4 2.4 GHz Radio Frequency (RF) emitting/receiving chip.

  There are several kind of distributed units (DSUs, also named PODs) :

- The "Stim/Bio" DSUs allows for both the stimulation of muscles for movement generation and for the acquisition of EMG signals for biofeedback purpose. They are made of two electronic boards :
  - A generic board embeds the housekeeping hardware and software to provide the electrical supply (battery), switches and LEDs, and a physical network transceiver. The protocol stack and the requests (stimulation and acquisition) processing are handled by a Freescale MC1322X micro-controller.
  - A daughter board executes the stimulation patterns or the acquisition requests through two input/output analogue channels.
- The "universal" DSUs use the same generic board, but embeds in its daughter board specific sensors. For example, IMUs or goniometers are planned to be embedded in a near future.

The communication architecture uses a specifically designed 3-layer protocol stack, compliant with the reduced OSI model. These layers are the Application layer, the MAC layer and the Physical layer. The physical layer manages the wireless medium. The MAC layer ensures a deterministic medium sharing, so that only one unit can speak at a time (no collision), to provide known and bounded response times. It allows for either a single (unicast) or a group of target DSUs (multicast) by using an individual or group identifier (Godary-Dejean et al., 2011).

These various tasks run on the μcontrollers under control of a CMX real-time multi-tasking executive.

3 SIMULATION SETUP

The long term objective is the safe and efficient implementation of motion control using FES feedback loops. As the analysis of such complex hybrid system is out of scope of pure theoretic analysis, we need a software tool to help for the design, sizing, analysis and tuning of the components and of their interaction, w.r.t. specified performance and safety levels.

More precisely, we need that the system uses an open source framework and run in real-time on stock

[1] http://www.vivaltis.com/

PCs. It must easily evolve from functional simulation to HIL to cover the different phases of software development. Hence, according to the design current phase, some simulated components might use quite simple models while others should be more detailed. For example, a preliminary analysis of the control loops sampling rate do not need a very detailed physical model of the communication medium (which could be added later on for refined tuning).

3.1 Simulation architecture

Processes and threads Based on previous experience the simulation software is designed as a set of Unix processes and Posix threads running under Linux on stock multi-core PCs. The first version of the simulator is a pure software which does not integrate any piece of real hardware, and calculations are performed with the usual floating point capabilities of the PC.

However, to mimic the real distributed architecture, it is split in 3 processes so that it can be run on a multi-core PC (see Figure 3), for example one core simulates the controller and each DSU is simulated on another core :

- the "Controller" process executes the FES controllers running on the control chip under supervision of a multitasks O.S.; it also simulates the network transceiver through a simplified model of the communication stack;
- the "Medium" process is used to simulate the physical features of the wireless link;
- the "DSU" process handles models of the Stim and Acquisition DSU fonctions in dedicated threads. It also handles the computation thread running the numerical integrator.

Using the (non portable) *pthread_attr_setaffinity_np* CPU allocation statement, each of these process can be run on a different core. In the future these processes might be easily allocated on different chips connected by the real transceivers to enhance the simulation framework towards an HIL simulation.

Communication and synchronization A simplified version of the communication protocol stack is implemented in dedicated communication threads. The data exchanged between the controller and the DSUs are encapsulated in packets whose structure is similar to those handled by the real network. In the simulation, the packets are handled in two steps by UDP Unix sockets (Figure 4) :

- the packet is first transmitted using a point-to-point UDP socket from the communication manager of the controller (or of the Pod) to the medium process receiving thread;
- in the current version, the model of the medium just add a constant delay to the packet transmission, and data loss are handled by a simple random process.
- after being delayed, the packet is sent to all receivers using a diffusion UDP socket (using the "so_broadcast" parameter of UDP sockets). This feature allows for sending data with a unique time stamp to groups of Pods.

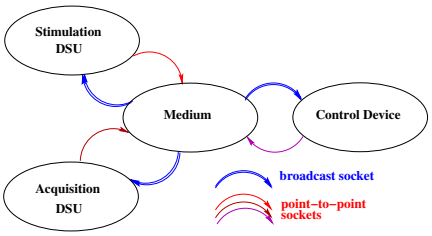

Figure 4: Communication sockets

Timing values All threads in the simulation can be unlocked either by signals from clocks (e.g. for periodic control tasks), by blocking waiting messages on an incoming socket or by semaphores signaled by communication requests sent by another thread. The threads are scheduled by the Linux kernel, set with the RT_FIFO real-time mode, according to their priorities and locks. However stock Linux kernels sometimes provides inconsistent behaviors, although the simulation works coherently with kernel compiled with real-time options, i.e. with high resolution timers and low latency preemption model under superuser privilege.

The transmission delays on the wireless links are simulated by pure delays in the medium component. Probes using functions of the Posix real-time library can be inserted anywhere in the threads to measure, e.g., communication delays between two tags or to record missed deadlines.

Solver triggering The numerical integration thread is a special case. Indeed this thread is not the ghost of a real time task existing in the real implementation.

In real life the state of the physical process naturally evolves in parallel with the numerical execution of the control and communication systems. The time scales of the continuous and numerical entities are meshed at some meeting instants at the occurence of different kind of events :

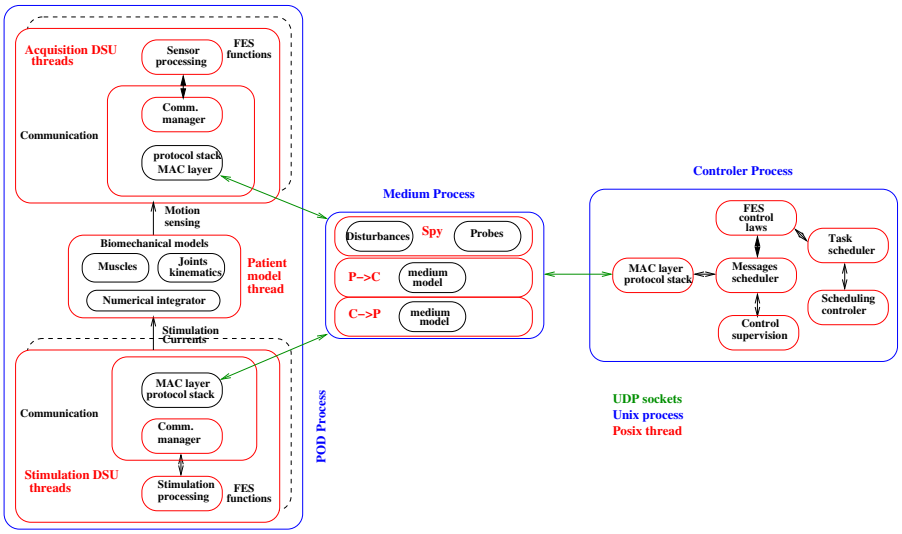

Figure 3: Simulation architecture

- when a control system requests an updated sensor value;
- when a control system sends a new control value;
- when the simulation system needs a snapshot of the controlled process and of its environment;
- when the continuous process raises an event (e.g., via an asynchronous sensor or crossing a model discontinuity).

The integration thread is triggered by any of these events. At each triggering event the integrator is first run from the last interruption until the current time to update the state of the continuous process, then it delivers the new state or accept new control inputs.

In practice, for the current set-up the simulation runs very comfortably faster than real-time on a modern laptop (e.g., 100 times faster on a i7 CPU). Computation times and overruns can be easily checked by software probes and reported to provide information and warnings about the timing behavior and quality of the simulation.

Note that, if the numerical integration cost becomes prohibitive to run the simulation on a single CPU w.r.t. real-time, the model can be split over several CPUs. Slackened synchronization between decoupled sub-models allows for significant speed-ups while keeping the integration errors under control (Ben Khaled-El Feki, 2014).

3.2 Continuous plant and numerical integration

Continuous model The model of the continuous plant is usually given as a set of ordinary differential equations (ODEs), differential algebraic equations (DAEs) or partial derivative equations (PDEs), where the continuous (physical) time is an independent variable.

Building high fidelity system-level models of complex systems, like mechatronic systems and even more systems involving livings, is a challenging duty. One problem is the diversity of modeling and simulation environments used by the various involved multi-disciplinary teams. Particular environments are preferred for a specific use due to distinctive strengths (modeling language, libraries, solvers, cost, etc.), for example the automatic derivation of the kinematic and dynamic equations of a multi-body system can be a distinctive feature.

Perfect modeling must be forgot. A model is always an approximation of reality, both in its structure and in its parameters values. A problem for simulation, due to both model complexity and fast system dynamics, is the prohibitive CPU times which can be observed when high-fidelity models are run, therefore preventing any real-time simulation. It is needed to find a satisfactory trade-off between the simulation speed and precision. A good model captures the essential behavior of the physical plant. Therefore, according to its user and objectives (e.g. an control scientist dealing with gains tuning, or a therapist willing to choose the stimulation frequency), different models of a given physical plant can be chosen.

Building a high-fidelity model needs precise knowledge about the plant itself, here coming from bio-mechanics and neuroscience. The simulation setup has been tested using the model of a human knee controlled by electrostimulation, a topic for which the Demar team developed a strong experience over years. The main structure and parameters of the model were mainly taken from (Benoussaad, 2009) for artificially excited muscles and from (McFaull and Lamontagne, 1998) for the joint non-linear parameters.

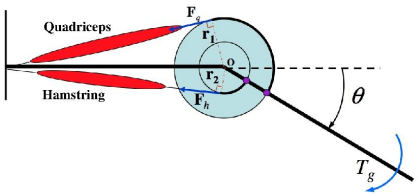

Figure 5: Simplified knee kinematics

It is given as a small set of non-linear ODEs. A contractile element (CE) dynamics is given by (Benoussaad, 2009) :

$$\begin{aligned} \dot{K}_c &= -K_c|u| + \alpha K_{cm} Fl_c(\epsilon_c)|u|_+ - K_c|\dot{\epsilon}_c| \\ \dot{F}_c &= -F_c|u| + \alpha F_{cm} Fl_c(\epsilon_c)|u|_+ - F_c|\dot{\epsilon}_c| + L_{c0} K_c \dot{\epsilon_c} \end{aligned} \quad (1)$$

where K_c and F_c are the stiffness and force developed by the CE, K_{cm} and F_{cm} their maximum values, u and $|u|_+$ chemical control variables coming from an activation model, $Fl_c(\epsilon_c)$ the force/length relation of the CE, α the recruitment rate, and $\dot{\epsilon}_c$ the contraction speed of the CE.

The numerical integration of the later variable, given by

$$\dot{\epsilon_c} = \frac{k_s L_0 \dot{\epsilon} + F_c|u| - \alpha F_{cm} Fl_c(\epsilon_c)|u|_+}{k_s L_{c0} + K_c L_{c0} - s_v F_c} \quad (2)$$

needs a special attention as it involves the *sign* function :

$$s_v = \text{sign}(k_s L_0 \dot{\epsilon} + F_c|u| - \alpha F_{cm} Fl_c(\epsilon_c)|u|_+) \quad (3)$$

Correctly handling this non-linearity needs an integrator enabled with a *root-finding* capability to provide a fast and stable integration (see next section).

The two muscles involved in the knee flexion/extension, i.e. the quadriceps and hamstring, have similar models with different parameters and are actuated by their own electrode. For our purpose, the knee joint is considered to be a perfect cylindrical d.o.f. leading the usual equation of motion (Figure 5

$$\ddot{\Theta} = \frac{1}{J}(F_2 r_2 - F_1 r_1 - MgL_{og} \sin(\Theta - \Theta_0) - B\dot{\Theta} + T_e) \quad (4)$$

where F_i is the force induced by muscle i, B a non-linear damping and T_e a passive elasticity torque.

Finally the model is encoded for integration as a set of 8 first order non-linear ODEs, three for each muscle and two for the joint dynamics.

Numerical integration The non-linear model of a robot (or of most mechatronic systems involving mechanical and electrical components) is usually described by a set of ordinary differential equations (ODEs) (or sometimes by a set of differential-algebraic equations (DAEs)) (Arnold et al., 2011). There already exist a broad family of algorithms and corresponding software used to numerically solve such a set of ODEs.

With most existing integration packages, the end-user's choices can be :

- set the value of a fixed-step integrator, leading to a predictable number of steps and computing time, but the precision cannot be controlled and may lead to false results;
- specify the precision of the integration and delegate the adaptive step management to the integrator, but in that case the computing time cannot be predicted.

With an adaptive step integration method the controlled variable is the precision (or more exactly the estimate of the integration error). The open-source *LSODAR* package we have elected (Hindmarsh and Petzold, 1995) for this experiment uses a variable step, variable number of steps integration algorithm. It is suitable for both stiff and non-stiff systems, and automatically selects between non-stiff and stiff methods.

It is associated with a root-finding stopping criteria : when it detects a sign change in a particular function of the states (root detection), it runs a root-finding algorithm to find the exact time point at which the sign change occurred. The later feature is essential to correctly integrate continuous plants with discontinuities, i.e. where the values of some states or parameters, or even the model structure itself, suddenly change where a given threshold is crossed (zero-crossing). It is for example the case for models of impacts, dry friction, and here for muscles contraction (due to eq. (3)).

3.3 Control design

To generate movement for a joint, two muscles (an agonist and antagonist pair) must be stimulated. For the knee these two muscles are the quadriceps on the anterior face (extension actuator) and the hamstring on the posterior face (flexion actuator). In a first simple

control approach, the two muscles will be actuated in mutual exclusion according to the flexion and extension phases of the movements, see Figure 6.

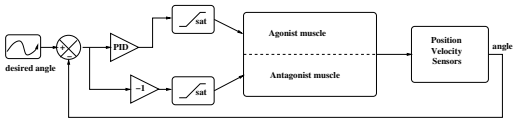

Figure 6: Control scheme

As usual, the first control algorithm to be considered is a PID, a simple controller able to control many single input/single output (SISO) systems through a very simple modeling and tuning effort. Here it is discretized using a backward difference method (with T_s the sampling period), yielding

$$u_k = u_{k-1} + K[e_k - e_{k-1}] + \frac{K \cdot T_s}{T_i} e_k + \frac{K \cdot T_d}{T_s}[e_k - 2e_{k-1} + e_{k-2}] \quad (5)$$

4 PRELIMINARY EXPERIMENTS

In this preliminary version, the simulation system is a "Model-in-the-loop" simulation, where all the objects are modeled, the physical plant state is evaluated by a numerical integrator and the control functions are encoded in plain C and executed using the floating point features of a modern laptop. Anyway, it goes beyond usual simulations with Scicos or Simulink as it handles quite precisely some implementation and scheduling related features such as tasks and events interleaving, or computation and communication induces delays. Hence it can be already used to explore the possible performance of FES feedback loops according to various scheduling parameters.

4.1 PID tuning and sampling rate

A simple PID controller has been setup for preliminary tests of the simulation platform. Indeed a PID with fixed gains would perform uniformly for a linear system, which is not the case for the knee joint, where even a simplified model shows several sources of non-linearities. Hence the PID has been empirically tuned to provide acceptable responses for the whole reachable motion amplitude.

Initially simulations were made with a 40 msec sampling rate, i.e. the one used by the marketed version of the Phenix system. In fact, the end-to-end communication time (including control tasks execution and messages transmission and acknowledgment) between the controller and the DSUs is lower than 5 msecs (Figure 7b). Lowering the sampling rate to 5 msecs shows an improved performance of

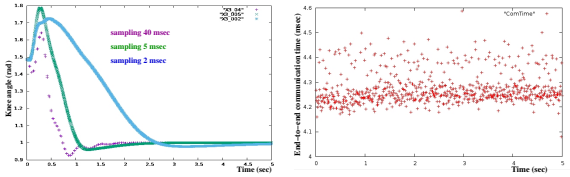

Figure 7: a) Knee position – b) End-to-end delay

the joint controller (Figure 7a). Note that, as the network model is included in the simulation, sampling rates smaller than 5 msec induce data loss and poor control performance. Hence the real-time simulator allows for a fine tuning of control gains w.r.t the execution resources at hand, far beyond the usually recommended sampling rates between 3 and 10 times the Shannon sampling rate.

4.2 Network protocol enhancements

The current communication threads do not implement the production code of the full protocol stacks, but only its main features. The simulated transceiver encapsulates data in frames according to the STIMAP protocol (Godary-Dejean et al., 2011) used to connect the controller and DSUs. The frames are further sent/received over UDP sockets. Several versions of the communication frames were tested, and their impact on the feedback control performance and on the communication load was evaluated (see Figure 8).

Single receiver (Production version) (in red) A data frame issued by the controller contains the address of a single receiving/answering DSU, and the payload only contains the stimulation parameters for this particular DSU (red plot, 6 ms average delay);

Single receiver with optimization (in green) The optimization uses the mutual exclusion between the agonist/antagonist muscle and send only frames to the active muscle in the current motion phase. However some safety messages are sent (at a slower rate compared with control messages) to keep alive the DSU of the passive muscle, leading to irregular timing patterns (green plot, 75% of samples near 3 ms and 25% of samples near 6 ms);

Multiple receivers (in blue) Messages are addressed to a group of DSUs, with a payload made of the concatenation of the individual stimulation parameters: the frame is longer than a single receiver frame but shorter than the concatenation of the single receiver frames, thus saving communication bandwidth (blue plot, 4.5 ms average delay);

The observation of the simulations traces show that using optimized versions of the protocol may significantly save bandwidth and decrease the end-to-end

communication delay between sensing and actuation, leading to improvements in the performance and robustness of the FES control law. Such results are useful to motivate (even costly) modifications of the production hardware and software.

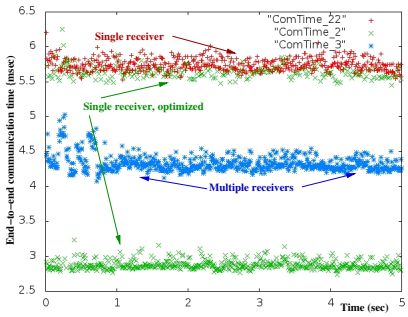

Figure 8: End-to-end communication time

5 SUMMARY AND FUTURE DEVELOPMENTS

Realistic simulations are effective tools to design and tune complex systems whose analysis cannot be provided only by theory. Several simulation steps can be explored, from simple functional analysis until HIL, to design, test, tune and validate both the single components of the system and their interactions in a distributed architecture. Simulations are precious, as they allow for non-destructive trials, anticipation for future technological development at lowered cost and exploration of a system behavior until its limits with no danger.

For the particular setup presented here, it is expected that such simulator will provide inputs in two main directions. Firstly it allows for preliminary testing new FES protocols prior to experiments with patients, and may help for writing the ethical protocols needed for any experiments involving livings. Secondly it can be used for preliminary evaluation of new technologies or implementations, without costly reworking of existing electronic chips or certified components.

The simulation software is open, so that enhancements w.r.t. to the original release can be added upon request of various designers and to fulfill various design objectives. For example, some future enhancements may be :

- Development and test of feedback schedulers for automatic adaptation of the CPU and communication loads in complex FES applications such as walking;
- Refinement of the physical communication medium model, so that the wireless link fading due to of obstacles can be evaluated, leading to the design of a feedback control of the emission power of the transceivers;
- Integration of others continuous plants. The dynamic model of a human hand with 23 d.o.f. is currently developed to provide a complex case for numerical integration and control coordination;
- Progressive integration of the production code of some components, e.g., the full protocol stack, and of some real components, e.g., wireless transceivers between simulated nodes, to incrementally provide HIL simulation setups.

As for any simulation, a crucial problem is the calibration of the models and the validation of the overall simulation system. This process needs, at some points, comparisons with observations of real experiments, which is costly but necessary. Anyway this is needed to gain the confidence of the various users of the system.

REFERENCES

Arnold, M., Burgermeister, B., Führer, C., Hippmann, G., and Rill, G. (2011). Numerical methods in vehicle system dynamics: State of the art and current developments. *Vehicle System Dynamics*.

Ben Khaled-El Feki, A. (2014). *Distributed real-time simulation of numerical models: application to powertrain*. PhD thesis, EEATS, Automatique Productique, Université de Grenoble. `http://www.theses.fr/2014GRENT033`.

Benoussaad, M. (2009). *Protocole d'identification sous FES et synthèse des séquences de stimulation chez le blessé médullaire*. PhD thesis, Univ. Montpellier II.

Godary-Dejean, K., Andreu, D., and Romain, R. (2011). Temporal bounds verification of the STIMAP protocol. In *RTNS 2011 : 19th International Conference on Real-Time and Network Systems*, page 10, Nantes, France.

Hindmarsh, A. C. and Petzold, L. R. (1995). Algorithms and software for Ordinary Differential Equations and Differential-Algebraic Equations, part II: Higher-order methods and software packages. *Comput. Phys.*, 9:148–155.

McFaull, S. R. and Lamontagne, M. (1998). In vivo measurement of the passive viscoelastic properties of the human knee joint. *Human Movement Science*, 17(2):139 – 165.

Zhang, Q., Hayashibe, M., and Azevedo Coste, C. (2013). Evoked Electromyography-Based Closed-Loop Torque Control in Functional Electrical Stimulation. *IEEE Transactions on Biomedical Engineering*, 60(8):2299–2307.